

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	

(1) 62 boo\$readprompt - prompt and read input string

```

0000 1      .title CONIO - console input output routines
0000 2      .ident /V04-000/
0000 3
0000 4
0000 5 *****
0000 6
0000 7      * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      * ALL RIGHTS RESERVED.
0000 10
0000 11      * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12      * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13      * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14      * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15      * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16      * TRANSFERRED.
0000 17
0000 18      * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19      * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20      * CORPORATION.
0000 21
0000 22      * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23      * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28      Facility: system bootstrapping
0000 29
0000 30      Abstract: CONIO provides basic console read, readprompt and write facilities.
0000 31
0000 32      Author: Richard I. Hustvedt, creation date: 27-apr-1978
0000 33
0000 34      Modified by:
0000 35
0000 36      V03-002 WHM0001      Bill Matthews      9-Jul-84
0000 37
0000 38      Add generalized alternate console terminal support
0000 39
0000 40      V03-001 DNC0001      David N. Cutler      29-Dec-83
0000 41
0000 42      Add support for QVSS as the console terminal on MicroVax I.
0000 43
0000 44      Include files:
0000 45
0000 46
0000 47      $prdef      ; define processor registers
0000 48      $ssdef      ; define status code values
0000 49
0000 50
0000 51      Equated symbols:
0000 52
0000 53
0000000D 0000 54      cr      = 13      ; character code for carriage return
0000000A 0000 55      lf      = 10      ; character code for line feed
00000015 0000 56      control_u = 21      ; character code for control-u
00000013 0000 57      control_s = 19      ; control s (xoff)

```


CONIO
V04-000

- console input output routines K 4

15-SEP-1984 23:48:08 VAX/VMS Macro V04-00
4-SEP-1984 23:03:51 [BOOTS.SRC]CONIO.MAR;1

Page 2
(1)

00000011 0000 58
0000007F 0000 59
00000000 0000 60

control_q = 17
rubout = 127
v_rub = 0

; control q (xon)
; character code for rubout
; rubout sequence in progress

```
0000 62 .sbtll boo$readprompt - prompt and read input string
0000 63 :+
0000 64 :
0000 65 : boo$readprompt outputs the specified asciz prompt string on the
0000 66 : console terminal then checks the count of characters to be read.
0000 67 : If zero it exits, otherwise it reads the console terminal until
0000 68 : either a carriage return is encountered or the character count
0000 69 : is satisfied. The specified buffer is filled with an asciz
0000 70 : string containing the characters read but not including the
0000 71 : terminating carriage return.
0000 72 : Calling sequence:
0000 73 :
0000 74 : callx arglist,boo$readprompt
0000 75 :
0000 76 : Input parameters:
0000 77 :
0000 78 : prompt(ap) - address of asciz prompt string
00000004 0000 79 : prompt = 4
0000 80 :
00000008 0000 81 : size(ap) - maximum length of input string
0000 82 : size = 8
0000 83 : note: if size is zero, then nothing is read
0000 84 : and only the prompt string is written.
0000 85 :
0000000C 0000 86 : buf(ap) - address of input buffer
0000 87 : buf = 12
0000 88 :
0000 89 :
0000 90 : Output parameters:
0000 91 :
0000 92 : r0 - completion status code (always ss$_normal)
0000 93 :
0000 94 : Buffer located by buf(ap) will be filled with the string
0000 95 : read as an asciz string.
0000 96 :
0000 97 :
00000000 0000 98 .psect $conio,byte
0000 99 .entry boo$readprompt,^m<r2,r4,r8,r9>
58 04 AC D0 0002 100 10$: movl prompt(ap),r8 ;get prompt string address
50 54 D4 0006 101 : clrl r4 ;clear control flags
50 88 9A 0008 102 20$: movzbl (r8)+,r0 ;get next output character
0075 05 13 000B 103 : beql 30$ ;if eql none
0075 30 000D 104 : bsbw con$putchar ;output character
F6 11 0010 105 : brb 20$ ;
52 08 AC 9A 0012 106 :
52 60 13 0016 107 30$: movzbl size(ap),r2 ;maximum number of characters to read
59 0C AC D0 0018 108 : beql 120$ ;if eql none
59 89 94 001C 109 : movl buf(ap),r9 ;set address of input buffer
02 52 F5 001E 110 : clrb (r9)+ ;initialize string count
42 11 0021 111 : sobgtr r2,40$ ;decrement and test character count
0023 112 : brb 110$ ;end of read
008F 30 0023 113 :
58 50 80 8F 8B 0026 114 40$: bsbw con$getchar ;get a character
58 7F 8F 91 002B 115 : bicb3 #^x80,r0,r8 ;clear parity bit
58 11 12 002F 116 : cmpb #rubout,r8 ;rubout?
58 79 9A 0031 117 : bneq 80$ ;if neq no
58 79 9A 0031 118 : movzbl -(r9),r8 ;get character to rubout
```



```
02 54 DC 13 0034 119 beql 30$ ;if eqi none
00 E2 0036 120 bbss #v_rub,r4,70$ ;set start of rubout sequence
40 10 003A 121 bsbb outbslsh ;output back slash
44 10 003C 122 70$: bsbb outr8 ;output rubbed out character
52 D6 003E 123 incl r2 ;adjust remaining character count
E1 11 0040 124 brb 40$ ;
0042 125
02 54 00 E5 0042 126 80$: bbcc #v_rub,r4,90$ ;terminate rubout sequence
34 10 0046 127 bsbb outbslsh ;output backslash
58 15 91 0048 128 90$: cmpb #control_u,r8 ;control u?
B5 13 004B 129 beql 10$ ;if eqi yes
03 58 06 E1 004D 130 bbc #6,r8,100$ ;if clr, then graphic
58 20 8A 0051 131 bicb #32,r8 ;convert to upper case
58 0D 91 0054 132 100$: cmpb #cr,r8 ;carriage return?
0C 13 0057 133 beql 110$ ;if eqi yes
52 D5 0059 134 tstl r2 ;any space left in buffer?
C6 13 005B 135 beql 40$ ;if eqi no
23 10 005D 136 bsbb outr8 ;echo character
89 58 90 005F 137 movb r8,(r9)+ ;buffer new character
BE 52 F4 0062 138 sobgeq r2,40$ ;reduce space remaining (always loop)
0065 139
58 0D 9A 0065 140 110$: movzbl #cr,r8 ;set carriage return character
1B 10 0068 141 bsbb con$putchar ;
50 0A 9A 006A 142 movzbl #lf,r0 ;yes send line feed also
16 10 006D 143 bsbb con$putchar ;output character in r0
OC BC 59 OC AC C2 006F 144 subl buf(ap),r9 ;compute character count + 1
59 01 83 0073 145 subb3 #1,r9,@buf(ap) ;set actual character count
50 01 3C 0078 146 120$: movzwl #ss$_normal,r0 ;return normal completion status
04 007B 147 ret ;
007C 148
007C 149 outbslsh: ;output back slash
50 5C 8F 9A 007C 150 movzbl #^a%\%,r0 ;set character code
03 11 0080 151 brb con$putchar ;and output it
0082 152
50 58 9A 0082 153 outr8: movzbl r8,r0 ;get character to output
0085 154
0085 155 con$putchar:: ;output character in r0
0000 31 0085 156 brw w^10$ ;console terminal output vector
0088 157
1B 51 20 DB 0088 158 10$: mfpr #pr$_rxcs,r1 ;receiver ready?
51 07 E1 008B 159 bbc #7,r1,30$ ;if clr, receiver not ready
51 21 DB 008F 160 mfpr #pr$_rxdb,r1 ;read input character.
13 51 07 00 ED 0092 161 cmpzv #0,#7,r1,#control_s ;control-s?
11 12 0097 162 bneq 30$ ;if neq no
51 20 DB 0099 163 20$: mfpr #pr$_rxcs,r1 ;receiver ready?
F9 51 07 E1 009C 164 bbc #7,r1,20$ ;if clr, receiver not ready
51 21 DB 00A0 165 mfpr #pr$_rxdb,r1 ;read input character
11 51 07 00 ED 00A3 166 cmpzv #0,#7,r1,#control_q ;is it a control-q?
EF 12 00A8 167 bneq 20$ ;no, wait for another character.
51 22 DB 00AA 168 30$: mfpr #pr$_txcs,r1 ;transmitter done?
F9 51 07 E1 00AD 169 bbc #7,r1,30$ ;if clr, transmitter not done
23 50 DA 00B1 170 mtp r0,#pr$_txdb ;write output character
05 00B4 171 rsb ;return
00B5 172
00B5 173
00B5 174 con$getchar::
0000 31 00B5 175 brw w^10$ ;console terminal input vector
```

```

      00B8 176
F9 50 20 DB 00B8 177 10$: mfpr #pr$_rxcs,r0      ;receiver ready?
      50 07 E1 00BB 178      bbc #7,r0,10$      ;if clr, receiver not ready
      50 21 DB 00BF 179      mfpr #pr$_rxdb,r0    ;read input character
      05 00C2 180      rsb                      ;return
      00C3 181
      00C3 182 con$owncty::                      ;these routines are noops in SYSBOOT
      00C3 183 con$releasecty::
      05 00C3 184      rsb
      00C4 185
      00C4 186      .end

```


CONIO
Symbol table

- console input output routines

B 5

15-SEP-1984 23:48:08 VAX/VMS Macro V04-00
4-SEP-1984 23:03:51 [BOOTS.SRC]CONIO.MAR;1

Page 6
(1)

BOOSREADPROMPT	00000000	RG	02	OPS_CVTHB	= 000068FD
BUF	= 0000000C			OPS_CVTHD	= 0000F7FD
CON\$GETCHAR	000000B5	RG	02	OPS_CVTHF	= 0000F6FD
CON\$OWNCTY	000000C3	RG	02	OPS_CVTHG	= 000076FD
CON\$PUTCHAR	00000085	RG	02	OPS_CVTHL	= 00006AFD
CON\$RELEASECTY	000000C3	RG	02	OPS_CVTHW	= 000069FD
CONTROL_Q	= 00000011			OPS_CVTLD	= 0000006E
CONTROL_S	= 00000013			OPS_CVTLF	= 0000004E
CONTROL_U	= 00000015			OPS_CVTLG	= 00004EFD
CR	= 0000000D			OPS_CVTLH	= 00006EFD
LF	= 0000000A			OPS_CVTLP	= 000000F9
OPS_ACBD	= 0000006F			OPS_CVTPL	= 00000036
OPS_ACBF	= 0000004F			OPS_CVTPT	= 00000008
OPS_ACBG	= 00004FFD			OPS_CVTPT	= 00000024
OPS_ACBH	= 00006FFD			OPS_CVTRDL	= 0000006B
OPS_ADDD2	= 00000060			OPS_CVTRFL	= 0000004B
OPS_ADDD3	= 00000061			OPS_CVTRGL	= 00004BFD
OPS_ADDF2	= 00000040			OPS_CVTRHL	= 00006BFD
OPS_ADDF3	= 00000041			OPS_CVTSP	= 00000009
OPS_ADDG2	= 000040FD			OPS_CVTTP	= 00000026
OPS_ADDG3	= 000041FD			OPS_CVTWD	= 0000006D
OPS_ADDH2	= 000060FD			OPS_CVTWF	= 0000004D
OPS_ADDH3	= 000061FD			OPS_CVTWG	= 00004DFD
OPS_ADDP4	= 00000020			OPS_CVTWH	= 00006DFD
OPS_ADDP6	= 00000021			OPS_DIVD2	= 00000066
OPS_ASHP	= 000000F8			OPS_DIVD3	= 00000067
OPS_CLRD	= 0000007C			OPS_DIVF2	= 00000046
OPS_CLRF	= 000000D4			OPS_DIVF3	= 00000047
OPS_CLRG	= 0000007C			OPS_DIVG2	= 000046FD
OPS_CLRH	= 00007CFD			OPS_DIVG3	= 000047FD
OPS_CMPD	= 00000071			OPS_DIVH2	= 000066FD
OPS_CMPF	= 00000051			OPS_DIVH3	= 000067FD
OPS_CMPG	= 000051FD			OPS_DIVP	= 00000027
OPS_CMPH	= 000071FD			OPS_EDITPC	= 00000038
OPS_CMPP3	= 00000035			OPS_EMODD	= 00000074
OPS_CMPP4	= 00000037			OPS_EMODF	= 00000054
OPS_CRC	= 0000000B			OPS_EMODG	= 000054FD
OPS_CVTBD	= 0000006C			OPS_EMODH	= 000074FD
OPS_CVTBF	= 0000004C			OPS_MATCHC	= 00000039
OPS_CVTBG	= 00004CFD			OPS_MNEGD	= 00000072
OPS_CVTBH	= 00006CFD			OPS_MNEGF	= 00000052
OPS_CVTDB	= 00000068			OPS_MNEGG	= 000052FD
OPS_CVTDF	= 00000076			OPS_MNEGH	= 000072FD
OPS_CVTDH	= 000032FD			OPS_MOVD	= 00000070
OPS_CVTDL	= 0000006A			OPS_MOVF	= 00000050
OPS_CVTDW	= 00000069			OPS_MOVG	= 000050FD
OPS_CVTFB	= 00000048			OPS_MOVH	= 000070FD
OPS_CVTFD	= 00000056			OPS_MOVP	= 00000034
OPS_CVTFG	= 000099FD			OPS_MOVTC	= 0000002E
OPS_CVTFH	= 000098FD			OPS_MOVTUC	= 0000002F
OPS_CVTFL	= 0000004A			OPS_MULD2	= 00000064
OPS_CVTFW	= 00000049			OPS_MULD3	= 00000065
OPS_CVTGB	= 000048FD			OPS_MULF2	= 00000044
OPS_CVTGF	= 000033FD			OPS_MULF3	= 00000045
OPS_CVTGH	= 000056FD			OPS_MULG2	= 000044FD
OPS_CVTGL	= 00004AFD			OPS_MULG3	= 000045FD
OPS_CVTGW	= 000049FD			OPS_MULH2	= 000064FD

CONIO
Symbol table

- console input output routines

C 5

15-SEP-1984 23:48:08 VAX/VMS Macro V04-00
4-SEP-1984 23:03:51 [BOOTS.SRC]CONIO.MAR;1

Page 7
(1)

OPS_MULH3	=	000065FD
OPS_MULP	=	00000025
OPS_POLYD	=	00000075
OPS_POLYF	=	00000055
OPS_POLYG	=	000055FD
OPS_POLYH	=	000075FD
OPS_SCANC	=	0000002A
OPS_SKPC	=	0000003B
OPS_SPANC	=	0000002B
OPS_SUBD2	=	00000062
OPS_SUBD3	=	00000063
OPS_SUBF2	=	00000042
OPS_SUBF3	=	00000043
OPS_SUBG2	=	000042FD
OPS_SUBG3	=	000043FD
OPS_SUBH2	=	000062FD
OPS_SUBH3	=	000063FD
OPS_SBP4	=	00000022
OPS_SBP6	=	00000023
OPS_TSTD	=	00000073
OPS_TSTF	=	00000053
OPS_TSTG	=	000053FD
OPS_TSTH	=	000073FD
OUTBSLSH		0000007C R 02
OUTR8		00000082 R 02
PRS_RXCS	=	00000020
PRS_RXDB	=	00000021
PRS_TXCS	=	00000022
PRS_TXDB	=	00000023
PROMPT	=	00000004
RUBOUT	=	0000007F
SIZE	=	00000008
SS\$ NORMAL	=	00000001
V_RUB	=	00000000

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes													
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE			
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE			
\$CONIO	000000C4 (196.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE			

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	38	00:00:00.07	00:00:00.39
Command processing	133	00:00:00.79	00:00:04.76
Pass 1	458	00:00:12.51	00:00:24.90
Symbol table sort	0	00:00:01.33	00:00:02.51
Pass 2	54	00:00:03.68	00:00:06.87
Symbol table output	18	00:00:00.14	00:00:00.34

CONIO
VAX-11 Macro Run Statistics

- console input output routines

D 5

15-SEP-1984 23:48:08 VAX/VMS Macro V04-00
4-SEP-1984 23:03:51 [BOOTS.SRC]CONIO.MAR;1

Page 8
(1)

Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	705	00:00:18.55	00:00:39.80

The working set limit was 1650 pages.
59843 bytes (117 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 865 non-local and 14 local symbols.
2938 source lines were read in Pass 1, producing 16 object records in Pass 2.
136 pages of virtual memory were used to define 135 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	6

918 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CONIO/OBJ=OBJ\$:CONIO MASD\$:[EMULAT.SRC]MISSING/UPDATE=(MASD\$:[EMULAT.ENH]MISSING)+MASD\$:[BOOTS.SRC]CONIO/UPDATE=(MASD

0038 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY